Embedded Linux Conference Europe 2013

Going Linux on Massive Multicore

Marta Rybczyńska

24th October, 2013

Agenda

- Architecture
- Linux Port
  - Core
  - Peripherals
- Debugging
- Summary and Future Plans

Agenda

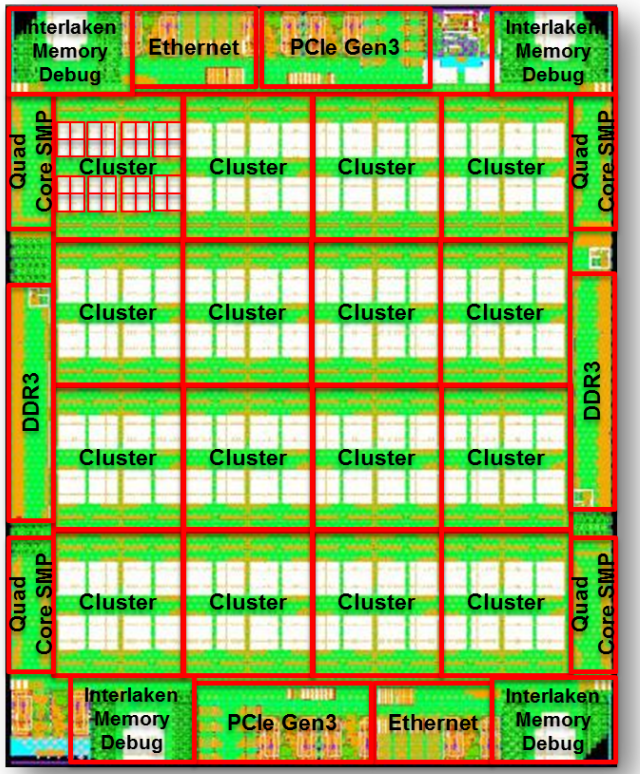
- **Architecture**
- Linux Port
  - Core
  - Peripherals
- Debugging
- Summary and Future Plans

First MPPA®-256 Chips with TSMC 28nm CMOS

256 Processing Engine cores + 32 Resource Management cores

- 256 (+32) user-programmable, generic cores
- Architecture and software scalability
- High processing performance
- High energy efficiency
- Execution predictability
- PCIe Gen3, Ethernet 10G, NoCX

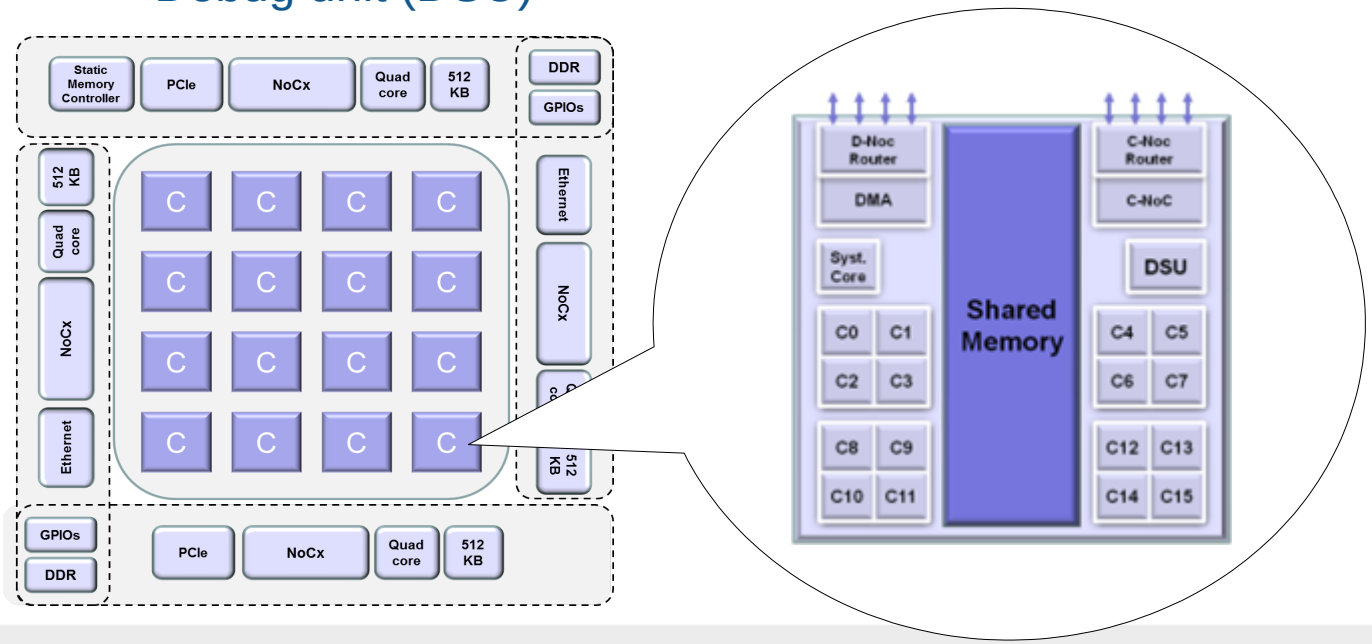
The MPPA-256 Processor (1)



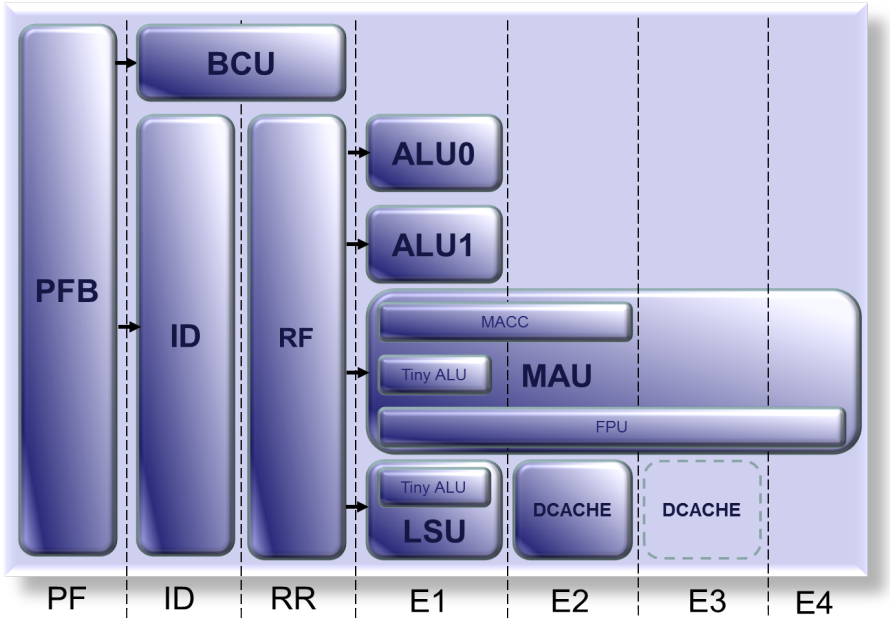
- 16 compute clusters
- 4 Input/Output clusters with rich peripheral access
  - DDR, PCI Express, Ethernet, Flash, GPIO, SPI, I2C etc
- Connected by a Network-on-Chip (NoC)

The MPPA-256 Processor (2)

- Compute cluster includes:
  - 16+1 cores
  - Shared memory
  - Network-on-Chip Interfaces
  - Debug unit (DSU)
- IO cluster includes:
  - 4 cores
  - Shared memory
  - Peripherals



The MPPA-256 Processor Core ISA



- Same on IO and compute cluster
- 5-issue Very Long Instruction Word (VLIW)
- 32/64-bit IEEE 754 floating point unit
- DSP instructions
- Advanced bitwise instructions
- Hardware loops
- MMU
- Idle modes

Kalray Software Development Kit for MPPA-256 (1)

- Standard Embedded C/C++ Programming
  - GCC, GNU binutils, newlib, uClibc, ...
- Simulators, Profilers, Debuggers & System Trace
  - GDB
  - Cycle-based simulator
  - Hardware System trace
- Operating systems & Device Drivers
  - NodeOS on the compute clusters
    - One thread per core
  - RTEMS port on the IO clusters
  - PCI Express driver for the Linux Host

Kalray Software Development Kit for MPPA-256 (2)

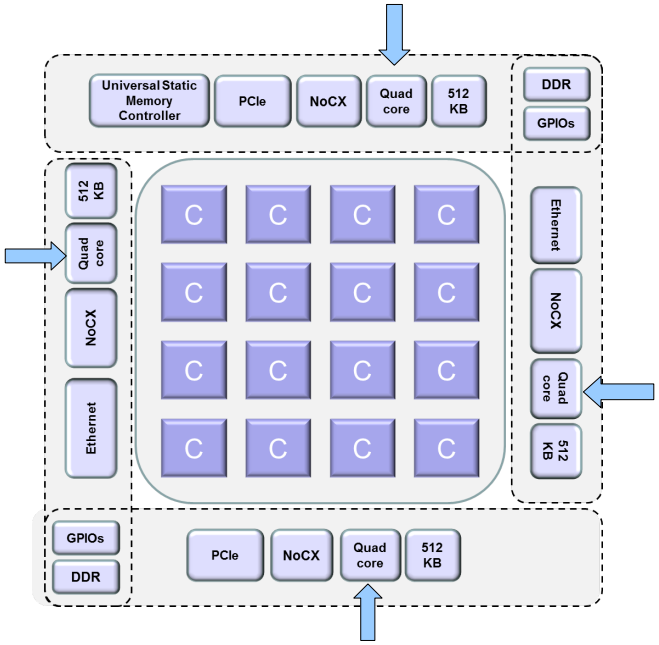
- Programming Models
  - Dataflow Programming
    - FPGA Style
  - POSIX-level Programming
    - DSP Style
  - Streaming Programming
    - GPU Style

The MPPA POSIX Programming Model

- POSIX-like process management
  - Spawn 16 processes from the IO cluster
  - Process execution on the 16 clusters start with main(argc,argv) and environment
  - On each cluster, support to up to 16 concurrent threads
- Inter-process communication (IPC)
  - POSIX file descriptor operations on 'NoC Connectors'
  - Extension to the PCIe interface with the 'PCIe Connectors'
  - Rich communication and synchronization
- Multi-threading inside clusters
  - Standard GCC/G++ OpenMP support
  - POSIX threads interface

The MPPA & Linux

- High-performance SMP on the IO clusters
- Device Drivers
  - SPI, I2C, GPIO (sensors, small peripherals)
- Full network stack
- Existing user libraries
- Rich configuration options



Agenda

- Architecture
- **Linux Port**
  - Core
  - Peripherals
- Debugging
- Summary and Future Plans

History

- 2.6.33-rc4
- 2.6.33
- 2.6.35
- 3.2
  - SMP, first specific drivers
- 3.10
  - tracing, MMU (in progress)

Features

- The kernel
  - Single core or SMP
  - Device-tree, generic headers
  - Supports userspace with FDPIC
  - Initial tracing support
- Drivers
  - Generic drivers tested
  - Specific drivers
    - PCI Express, console,...
- Userspace: buildroot-based with custom toolchain & uClibc

Agenda

- Architecture
- Linux Port
  - **Core**
  - Peripherals
- Debugging
- Summary and Future Plans

Optimization Case: Atomic Ops

- Single core case: disable/enable interrupts
  - Cheap on MPPA
- SMP case: compare-and-swap
  - Huge performance/code size impact
  - Last version uses implementation details of the caches and the write buffer
  - Return from experience
    - Improved atomic ops for the next version of the core

```
static inline void atomic_set(atomic_t *v, int i)
{
        __k1_atomic_visibility_pre();

        __k1_umem_write32((void *) &(v)->counter, i);
        __builtin_k1_fence();

        __k1_atomic_visibility_post(&(v)->counter);
}
```

Lessons Learned 1: Read the Specs Carefully (1)

- SMP version started deadlocking
- Spinlock ticket value showed corruption
- Debugging
  - Spinlock ordering problem?
    - Careful platform code analysis
  - Enabled spinlock debug
  - Detailed simulator trace analysis
- Reason

```
static inline
unsigned long __xchg_u32(volatile void *ptr, unsigned long x)
{
        unsigned long old;
        ...

        do {
                old = __k1_umem_read32((void *) ptr);
        } while (cmpxchg((unsigned int *) ptr, old, x) != old);

        return x; /* original */
        return old; /* fixed */
}
```

Lessons Learned 2: Use the GCC

- K1 core is a VLIW: multiple instructions (one bundle) per cycle
- High performance gain
  - GCC handles it well
  - Manual bundling OK for short code, hard for longer ones
- Result
  - Preferring built-ins over asm inlines
  - Less assembly in the code

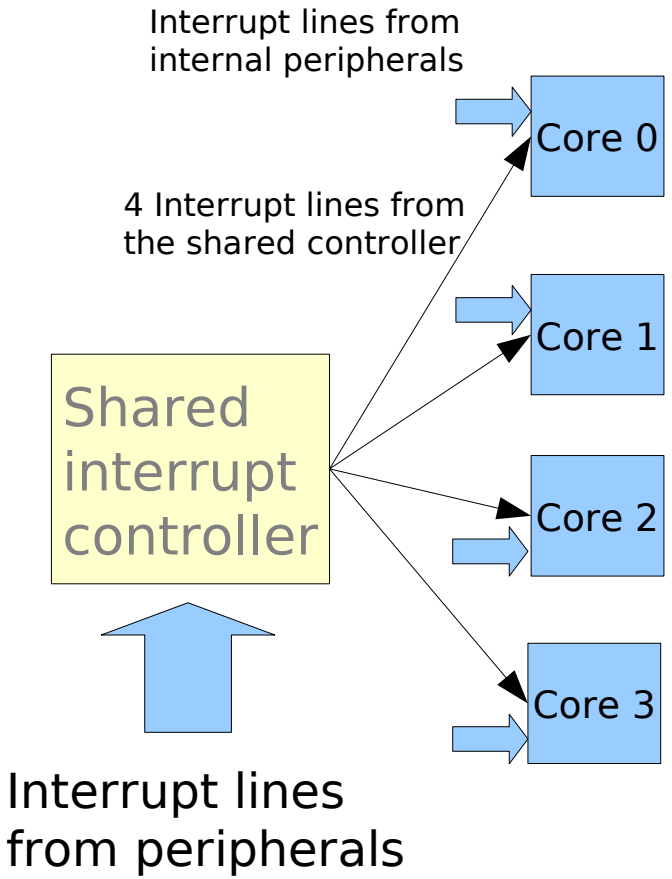
```
__mcount:
        add $r53 = $r33, 16
        copy $r40 = $r33
        get $r41 = $sr0
        ;;
#ifdef CONFIG_K1_TRACES
# Generate HW trace with 2x32 bit values
# args: r40, r41
__mcount_tracepoint:
        get $r38 = $pcr
        make $r35 = 0x1 ## tracepoint name
        make $r34 = 136
        ;;
        insf $r34 = $r35, 31, 16
        extfz $r38 = $r38, 15, 11
        ;;
        srl $r35 = $r35, 16
        insf $r34 = $r38, 12, 8
        ;;
        make $r33 = 0
        copy $r32 = $r40
        copy $r38 = $r41
        make $r40 = 1879588896
        ;;
        copy $r39 = $r33
        slld $r32:$r33 = $r32:$r33,16
        or $r36 = $r34, 5
        ;;
```

Agenda

- Architecture
- Linux Port
  - Core
  - **Peripherals**
- Debugging
- Summary and Future Plans

Interrupt Controllers

- Two-level interrupt control
  - One private to each core (timers, NoC)
  - Shared between 4 cores (PCI Express, Ethernet, other peripherals)
- Basic operations on the core interrupt controller
- Multiple devices on the same line
  - Configuration issue: configuring lanes for devices



Memory Areas

- Two areas
  - Big DDR
  - Small internal shared memory
- Different latency, optimization opportunity
- Shared memory is visible through PCI Express
- Solution:
  - Allocation in the DDR by default
  - Separate allocator for the shared memory

The PCI Express Driver (1)

- PCI Express is the main interface to the MPPA
  - Host (Linux) driver is ~10KLOC
  - Boot, synchronization, DMA, message passing,...
  - Two interfaces per MPPA (visible as two PCI Express devices)
- Host-side framework in Linux is mature
- Less standard on the device side
  - Code reuse (protocols)
  - First synchronization in the bootloader

The PCI Express Driver (2)

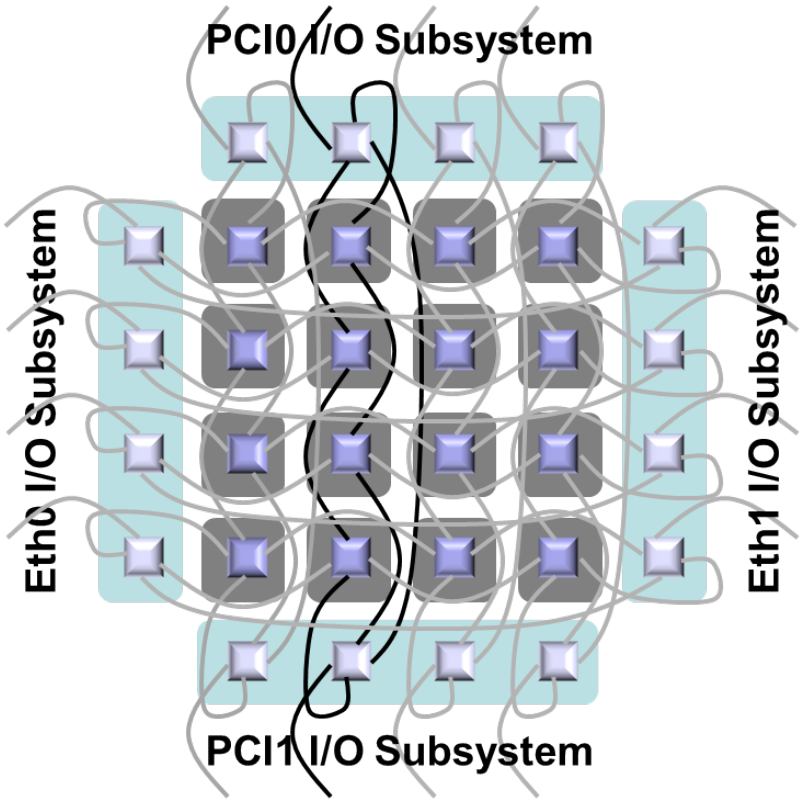
- PCI Express allows memory-mapping on Host of memory zones of the device
- Hardware resources
  - Doorbell registers (shared, in BAR0)
  - Shared memory accessible by the Host directly (BAR2)
  - Both DDR and shared memory accessible by DMA
- Cache effects
  - Shared zones negotiated (especially shared memory)

Boot

- Using custom bootloaders at the moment
- Boot of IO cluster by PCI Express
  - Initiated by the Host
  - First code in shared memory
  - Then complete image in shared memory+DDR
- Boot of clusters from the IO
  - Cluster executables in IO cluster memory
  - Also transferred by PCI Express

Network-on-Chip

- The way to communicate between clusters
  - High performance interface
  - Shared resources
- Used in different places
  - Boot, drivers in the kernel space
  - User code (IPC)
  - No Linux kernel API to reuse



The Ethernet Device Drivers

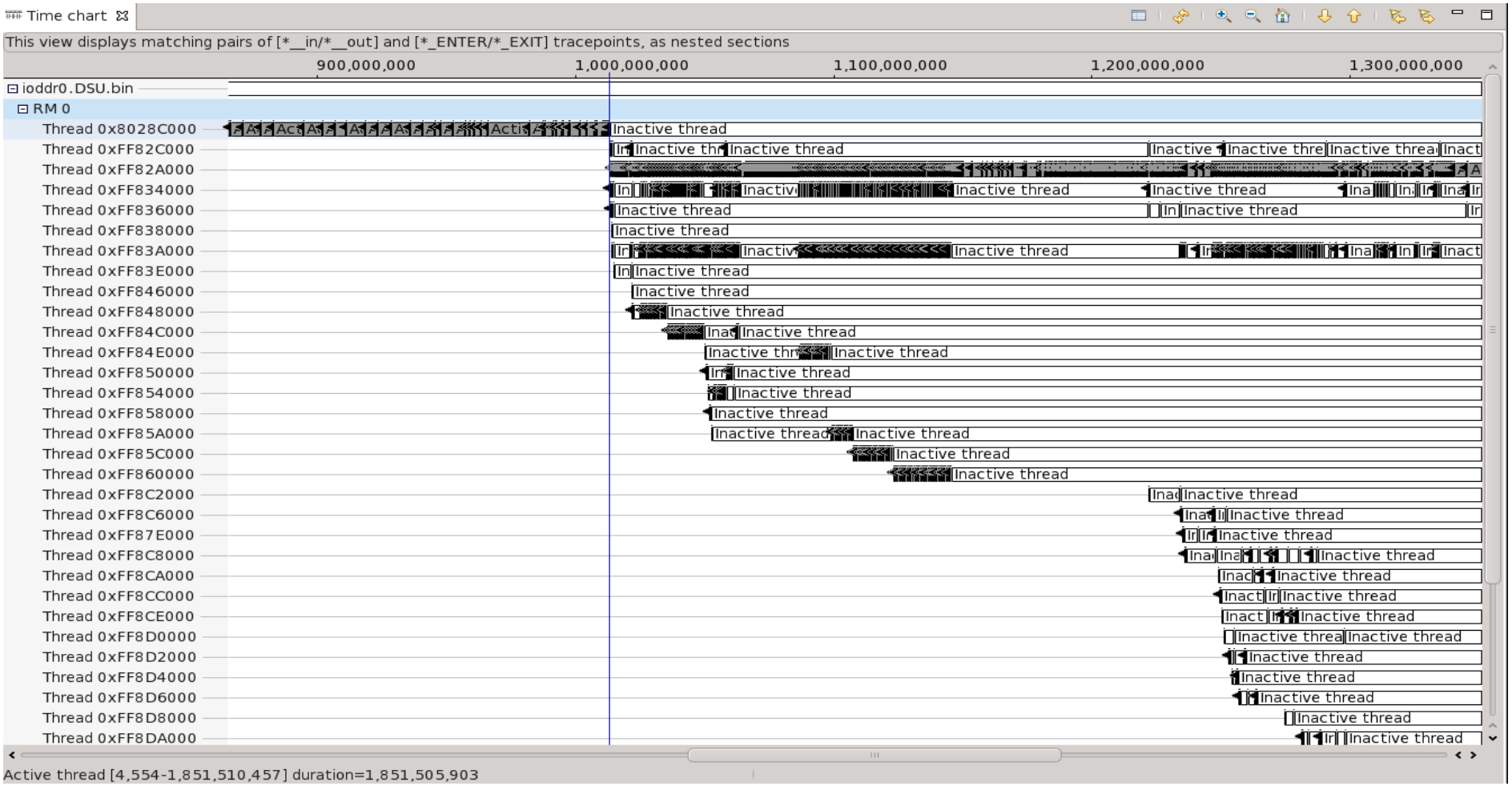
- Up to 8x10Gbit/s
  - MAC controller, PHY
  - Uses NoC
- Distributed network stack
  - Packet dispatch to different clusters
  - Potentially applications using different protocols in different clusters

Agenda

- Architecture
- Linux Port
  - Core
  - Peripherals
- **Debugging**
- Summary and Future Plans

KALRAY

Traces



KALRAY

printf(), printk() and GDB

```
IODDR0: 00: bootmem::__reserve nid=0 start=81303 end=81304 flags=1
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=4 [1 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81304 end=81304 flags=1
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=4 [1 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81304 end=81304 flags=1
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=4 [1 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81304 end=81304 flags=1
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=78 [1 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81304 end=81304 flags=1
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=2c [1 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81304 end=81304 flags=1
IODDR0: 00: Kernel command line: dhash_entries=1024 ihash_entries=1024 bootmem_debug=1 init=/sbin/ini
t lpj=4005057 CONSOLE=/dev/ttyS0 LD_DEBUG=all
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=4000 [4 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81304 end=81308 flags=1
IODDR0: 00: PID hash table entries: 4096 (order: 2, 16384 bytes)
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=1000 [1 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81308 end=81309 flags=1
IODDR0: 00: Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
IODDR0: 00: bootmem::alloc_bootmem_bdata nid=0 size=1000 [1 pages] align=20 goal=80000000 limit=0
IODDR0: 00: bootmem::__reserve nid=0 start=81309 end=8130a flags=1
IODDR0: 00: Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
IODDR0: 00: Sorting __ex_table...
IODDR0: 00: bootmem::free_all_bootmem_core nid=0 start=802e5 end=fffff
IODDR0: 00: bootmem::free_all_bootmem_core nid=0 released=7eceb
IODDR0: 00: totalram_pages = 519403
IODDR0: 00: reserved = 4143
IODDR0: 00: kernel code 80023000 - 80232430
IODDR0: 00: kernel data 80232430 - 802e4440
IODDR0: 00:
IODDR0: 00: Memory available: 2077612k/2097148k RAM,
IODDR0: 00: Kernel occupies 2961k (2109k kernel code, 712k data, 128k init)

IODDR0: 00: Reserved memory: 16572k
IODDR0: 00:
IODDR0: 00: SLUB: HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
IODDR0: 00: NR_IRQS:80
IODDR0: 00: K1 GIC driver, ID : 0, Version 0:0
IODDR0: 00: Time init done
IODDR0: 00: console [ttyS0] enabled
IODDR0: 00: Console: colour dummy device 80x25
IODDR0: 00: Calibrating delay loop (skipped) preset value.. 801.01 BogoMIPS (lpj=4005057)
IODDR0: 00: pid_max: default: 4096 minimum: 301
IODDR0: 00: Mount-cache hash table entries: 512
```

```
End of assembler dump.
(gdb) si
0x800fc58c in __mcount ()
(gdb)
0x800fc59c in __mcount ()
(gdb) bt
#0  0x800fc59c in __mcount ()
(gdb) s
Single stepping until exit from function __mcount,
which has no line number information.
__switch_to (prev=0xff828000, next=0xff8282a0) at arch/k1/kernel/switch.c:15
15      in arch/k1/kernel/switch.c
(gdb) s
__k1_mb () at arch/k1/kernel/switch.c:15
15      in arch/k1/kernel/switch.c
(gdb) bt
#0  __k1_mb () at arch/k1/kernel/switch.c:15
#1  __switch_to (prev=0xff828000, next=0xff8282a0) at arch/k1/kernel/switch.c:15
#2  0x802318e0 in context_switch (next=0xff8282a0, prev=0xff828000, rq=0x80293718) at kernel/sched/co
re.c:2018
#3  __schedule () at kernel/sched/core.c:3010
#4  0x802319b0 in sched_submit_work (tsk=0xff828000) at kernel/sched/core.c:3046
#5  schedule () at kernel/sched/core.c:3045
#6  0x8023006c in schedule_timeout (timeout=-8224096) at kernel/timer.c:1445
#7  0x80231488 in do_wait_for_common (state=-8224096, timeout=2147483647, action=<optimized out>, x=0
xff82be1c) at kernel/sched/core.c:3311
#8  __wait_for_common (state=-8224096, timeout=<optimized out>, action=<optimized out>, x=0xff82be1c)
 at kernel/sched/core.c:3329
#9  wait_for_common (x=0xff82be1c, timeout=2147483647, state=-8224096) at kernel/sched/core.c:3337
#10 0x80231a20 in wait_for_completion (x=0xff82be1c) at kernel/sched/core.c:3359
#11 0x8004a704 in kthread_create_on_node (threadfn=0x80052478 <smpboot_thread_fn>, data=0xff802210, n
ode=0, namefmt=0x80274951 "ksoftirqd/%u") at kernel/kthread.c:270
#12 0x8004a884 in kthread_create_on_cpu (threadfn=0x80052478 <smpboot_thread_fn>, data=0xff802210, cp
u=0, namefmt=<optimized out>) at kernel/kthread.c:335
#13 0x8005269c in __smpboot_create_thread (ht=0x80290bac, cpu=4286743200) at kernel/smpboot.c:180
#14 0x80052950 in __smpboot_create_thread (cpu=0, ht=0x80290bac) at kernel/smpboot.c:171
#15 smpboot_register_percpu_thread (plug_thread=0x80290bac) at kernel/smpboot.c:284
#16 0x80004e40 in spawn_ksoftirqd () at kernel/softirq.c:865
#17 0x800018e0 in do_one_initcall (fn=0x80004e28 <spawn_ksoftirqd>) at init/main.c:686
#18 0x80001a10 in do_pre_smp_initcalls () at init/main.c:787
#19 kernel_init_freeable () at init/main.c:874
#20 0x8022b2b4 in kernel_init (unused=<optimized out>) at init/main.c:813
#21 0x800233e4 in ret_from_kernel_thread ()
(gdb)
```

KALRAY

The Simulator

```
>k1-cluster --mcluster=ioddr -- vmlinux
Compiled-in FDT at 0x8001a0a0
Linux version 3.10.0+ (mrybczyn@doros) (gcc version 4.7.4 20130620 (prerelease) [Kalray Compiler unknown af8028d-dirty] (GCC) ) #1 SMP Fri Aug 23 13:42:41 CEST 2013
Cpu clock: 400MHz
setup_memory: Memory: 0x80000000-0x84000000
bootmem::init_bootmem_core nid=0 start=802d1 map=802d1 end=84000 mapsize=7a8
bootmem::mark_bootmem_node nid=0 start=802d1 end=84000 reserve=0 flags=0
bootmem::__free nid=0 start=802d1 end=84000
bootmem::mark_bootmem_node nid=0 start=802d1 end=802d2 reserve=1 flags=0
bootmem::__reserve nid=0 start=802d1 end=802d2 flags=0
Reserved - 0x83fff960-0x000006a0
bootmem::mark_bootmem_node nid=0 start=83fff end=84000 reserve=1 flags=0
bootmem::__reserve nid=0 start=83fff end=84000 flags=0
bootmem::alloc_bootmem_bdata nid=0 size=80000 [128 pages] align=20 goal=80000000 limit
bootmem::__reserve nid=0 start=802d2 end=80352 flags=1
bootmem::alloc_bootmem_bdata nid=0 size=8 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=80352 end=80353 flags=1
bootmem::alloc_bootmem_bdata nid=0 size=600 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=80353 end=80353 flags=1
bootmem::alloc_bootmem_bdata nid=0 size=4d [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=80353 end=80353 flags=1
bootmem::alloc_bootmem_bdata nid=0 size=4d [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=80353 end=80353 flags=1
bootmem::alloc_bootmem_bdata nid=0 size=1000 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=80353 end=80354 flags=1
bootmem::alloc_bootmem_bdata nid=0 size=1000 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=80354 end=80355 flags=1
bootmem::alloc_bootmem_bdata nid=0 size=20000 [32 pages] align=1000 goal=80000000 limi
bootmem::__reserve nid=0 start=80355 end=80375 flags=1
bootmem::mark_bootmem_node nid=0 start=8035a end=8035d reserve=0 flags=0
bootmem::__free nid=0 start=8035a end=8035d
bootmem::mark_bootmem_node nid=0 start=80362 end=80365 reserve=0 flags=0
bootmem::__free nid=0 start=80362 end=80365
bootmem::mark_bootmem_node nid=0 start=8036a end=8036d reserve=0 flags=0
bootmem::__free nid=0 start=8036a end=8036d
bootmem::mark_bootmem_node nid=0 start=80372 end=80375 reserve=0 flags=0
bootmem::__free nid=0 start=80372 end=80375
PERCPU: Embedded 5 pages/cpu @80355000 s6304 r0 d14176 u32768
bootmem::alloc_bootmem_bdata nid=0 size=4 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=8035a end=8035b flags=1
bootmem::alloc_bootmem_bdata nid=0 size=4 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=8035b end=8035b flags=1
bootmem::alloc_bootmem_bdata nid=0 size=10 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=8035b end=8035b flags=1
bootmem::alloc_bootmem_bdata nid=0 size=10 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=8035b end=8035b flags=1
bootmem::alloc_bootmem_bdata nid=0 size=78 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=8035b end=8035b flags=1
bootmem::alloc_bootmem_bdata nid=0 size=2c [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=8035b end=8035b flags=1
bootmem::mark_bootmem_node nid=0 start=80353 end=80353 reserve=0 flags=0
bootmem::__free nid=0 start=80353 end=80353
bootmem::mark_bootmem_node nid=0 start=80354 end=80354 reserve=0 flags=0
bootmem::__free nid=0 start=80354 end=80354
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 16256
Kernel command line: dhash_entries=1024 ihash_entries=1024 bootmem_debug=1 init=/init
bootmem::alloc_bootmem_bdata nid=0 size=400 [1 pages] align=20 goal=80000000 limit=0
bootmem::__reserve nid=0 start=8035b end=8035b flags=1
```

```
        2042427:        0x800cd6a0:     copy $r0(0x83813540) = $r10(0x83813540)
        2042427:        0x800cd6a0:     lw $r8(0x80005150) = 44[$r12(0x8002bfa0)] [V@ 0x8002bfcc ; P
        2042428:        0x800cd6a8:     lw $r10(0x8028a374) = 16[$r12(0x8002bfa0)] [V@ 0x8002bfb0 ; [
        2042429:        0x800cd6ac:     lw $r15(0x80016630) = 20[$r12(0x8002bfa0)] [V@ 0x8002bfb4 ; [
        2042430:        0x800cd6b0:     ld $r16r17(0x8026605480266048) = 24[$r12(0x8002bfa0)] [V@ 0x
        2042431:        0x800cd6b4:     set $ra(0x80005150) = $r8(0x80005150)
        2042431:        0x800cd6b4:     ld $r18r19(0x80315280) = 32[$r12(0x8002bfa0)] [V@ 0x8002bfc0
        2042432:        0x800cd6bc:     ret
        2042432:        0x800cd6bc:     add $r12(0x8002bfc8) = $r12(0x8002bfa0), 40
        2042432:        0x800cd6bc:     lw $r20(0x0) = 40[$r12(0x8002bfa0)] [V@ 0x8002bfc8 ; P@ 0x80
}}} __register_sysctl_paths
}}} register_sysctl_paths
}}} register_sysctl_table
        2042433:        0x80005150:     make $r0(0x0) = 0
        2042433:        0x80005150:     add $r12(0x8002bfd0) = $r12(0x8002bfc8), 8
        2042433:        0x80005150:     lw $r8(0x80001804) = 16[$r12(0x8002bfc8)] [V@ 0x8002bfd8 ; P
        2042434:        0x8000515c:     set $ra(0x80001804) = $r8(0x80001804)
        2042435:        0x80005160:     ret
}}} sysctl_init
}}} proc_sys_init
}}} proc_root_init
        2042436:        0x80001804:     call 2079364
{{{ rest_init
        2042437:        0x801fd288:     add $r12(0x8002bfc8) = $r12(0x8002bfd0), -8
        2042437:        0x801fd288:     make $r1(0x0) = 0
        2042437:        0x801fd288:     make $r2(0xa00) = 2560
        2042437:        0x801fd288:     make $r0(0x801fd308) = -2145398008
        2042438:        0x801fd29c:     get $r8(0x80001808) = $ra(0x80001808)
        2042439:        0x801fd2a0:     call -1882616
        2042439:        0x801fd2a0:     sw 16[$r12(0x8002bfc8)] = $r8(0x80001808) [V@ 0x8002bfd8 ; P
{{{ kernel_thread
        2042440:        0x800318a8:     copy $r3(0x0) = $r1(0x0)
        2042440:        0x800318a8:     copy $r4(0x801fd308) = $r0(0x801fd308)
        2042440:        0x800318a8:     or $r0(0x800b00) = $r2(0xa00), 8388864
        2042441:        0x800318b8:     copy $r2(0x0) = $r3(0x0)
        2042441:        0x800318b8:     make $r3(0x0) = 0
        2042441:        0x800318b8:     copy $r1(0x801fd308) = $r4(0x801fd308)
        2042442:        0x800318c4:     goto -652
        2042442:        0x800318c4:     copy $r4(0x0) = $r3(0x0)
{{{ do_fork
        2042443:        0x80031638:     get $r8(0x801fd2a8) = $ra(0x801fd2a8)
        2042443:        0x80031638:     add $r12(0x8002bfa0) = $r12(0x8002bfc8), -40
        2042444:        0x80031640:     and $r0(0x0) = $r0(0x800b00), 29, 28
        2042444:        0x80031640:     copy $r15(0x800b00) = $r0(0x800b00)
        2042444:        0x80031640:     sw 20[$r12(0x8002bfa0)] = $r15(0x80016630) [V@ 0x8002bfb4 ; [
```

Agenda

- Architecture
- Linux Port
  - Core
  - Peripherals
- Debugging
- **Summary and Future Plans**

Lessons Learned 3: What was Difficult

- Documentation missing
  - With great exceptions!
- Need to read other platform code
  - No examples (yet) for our architecture
- Reaching limits of the existing software
  - Not always a bug in the Linux port
- Trying to be mainline-compatible
- Initial port hard to split between developers
- Time-consuming, needed alternative for early testing (RTEMS port)

Lessons Learned 4: What was Easy

- Debugging
  - Mixed mode depending on the testcase: simulator, gdb, FPGA
  - Simulator trace postprocessing
- Generic headers cover a good part of the code
- Recently merged architectures give good examples

Future Plans

- Complete driver support (Ethernet, NoC, others...)
- Optimized MMU
- Replace generic implementations with optimized ones
- Public release
- Mainlining

Marta Rybczynska
marta.rybczynska@kalray.eu

http://www.kalray.eu